Word Embeddings with Limited Memory

词向量

Abstract

This paper studies the effect of limited precision data representation and computation on word embeddings. We present a systematic evaluation of word embeddings with limited memory and discuss methods that directly train the limited precision representation with limited memory. Our results show that it is possible to use and train an 8-bit fixed-point value for word embedding without loss of performance in word/phrase similarity and dependency parsing tasks.

本文研究了有限精度数据的表示和计算对词向量的影响。提出了一个用有限内存来评估词向量的系统，讨论用有限的内存来训练有限精度表示的方法。研究结果表明，通过一个8位的固定点值的词向量来使用训练，不会失去词/短语之间的相似性和在解析时它们之间的依赖。

1 Introduction

There is an accumulation of evidence that the use of dense distributional lexical representations, known as word embeddings, often supports better performance on a range of NLP tasks (Bengio et al., 2003; Turian et al., 2010; Collobert et al., 2011; Mikolov et al., 2013a; Mikolov et al., 2013b; Levy et al., 2015). Consequently, word embeddings have been commonly used in the last few years for lexical similarity tasks and as features in multiple, syntactic and semantic, NLP applications.

不断的证据表明，使用密集分布词汇表示（称为词向量）通常能更好的支持一系列NLP任务的性能（Bengio等人，2003; Turian等人，2010; Collobert等人， 2011; Mikolov等人，2013a; Mikolov等人，2013b; Levy等人，2015）。 因此，在过去几年中词向量被广泛用于词汇相似性任务，同时也应用于多特征，句法和语义，NLP等的应用中。

However, keeping embedding vectors for hundreds of thousands of words for repeated use could take its toll both on storing the word vectors on disk and, even more so, on loading them into memory. For example, for 1 million words, loading 200 dimensional vectors takes up to 1.6 GB memory on a 64-bit system. Considering applications that make use of billions of tokens and multiple languages, size issues impose signiﬁcant limitations on the practical use of word embeddings.

然而，词向量存储在磁盘上，甚至在将它们加载到存储器中时，要保持用于重复使用的数十万个字的词向量将付出代价。 例如，对于100万字，在64位系统上加载200维向量需要高达1.6 GB的内存。 考虑使用数十亿个tokens和多种语言的应用程序，内存大小问题对词向量的实际使用增加了显着的限制。

This paper presents the question of whether it is possible to signiﬁcantly reduce the memory needs for the use and training of word embeddings. Speciﬁcally, we ask “what is the impact of representing each dimension of a dense representation with signiﬁcantly fewer bits than the standard 64 bits?” Moreover, we investigate the possibility of directly training dense embedding vectors using signiﬁcantly fewer bits than typically used.

本文提出的问题是，是否能够显着减少使用和训练词向量的内存需求。具体来说，我们要求“如果使用更少的bits来表示每个维度的密集显示（应该是指词向量表示），产生的影响是否比标准64位少得多”？此外，我们研究使用比通常使用的位数少得多的位数作为词向量来直接训练密集的可能性。

The results we present are quite surprising. We show that it is possible to reduce the memory consumption by an order of magnitude both when word embeddings are being used and in training. In the ﬁrst case, as we show, simply truncating the resulting representations after training and using a smaller number of bits (as low as 4 bits per dimension) results in comparable performance to the use of 64 bits. Moreover, we provide two ways to train existing algorithms (Mikolov et al., 2013a; Mikolov et al., 2013b) when the memory is limited during training and show that, here, too, an order of magnitude saving in memory is possible without degrading performance. We conduct comprehensive experiments on existing word and phrase similarity and relatedness datasets as well as on dependency parsing, to evaluate these results. Our experiments show that, in all cases and without loss in performance, 8 bits can be used when the current standard is 64 and, in some cases, only 4 bits per dimension are sufﬁcient, reducing the amount of space required by a factor of 16. The truncated word embeddings are available from the papers web page at https://cogcomp.cs.illinois. edu /page/publication\_view/790.

我们的结果是相当令人惊讶的。当使用词向量和训练时，可以减少内存消耗的一个数量级。在第一种情况下，如我们所示，在训练之后简单的截断结果表示，并使用较少数量的位（每个维度低至4位）能够得到与使用64位相当的性能。此外，我们提供了两种训练现有算法的方法（Mikolov等人，2013a; Mikolov等人，2013b），当训练期间存储器有限时，不用降低性能来达到存储器中节省的数量级，也是可能的。我们对现有的单词与短语的相似性，和相关数据集以及依赖性解析进行了综合实验，以评估这些结果。我们的实验表明，在所有情况下，在没有性能损失的情况下，如果当前标准为64时，可以使用8位，在某些情况下，每维只有4位是足够的，从而将所需的空间量减少了16倍。

2 Related Work

If we consider traditional cluster encoded word representation, e.g., Brown clusters (Brown et al., 1992), it only uses a small number of bits to track the path on a hierarchical tree of word clusters to represent each word. In fact, word embedding generalized the idea of discrete clustering representation to continuous vector representation in language models, with the goal of improving the continuous word analogy prediction and generalization ability (Bengio et al., 2003; Mikolov et al., 2013a; Mikolov et al., 2013b). However, it has been proven that Brown clusters as discrete features are even better than continuous word embedding as features for named entity recognition tasks (Ratinov and Roth, 2009). Guo et al. (Guo et al., 2014) further tried to binarize embeddings using a threshold tuned for each dimension, and essentially used less than two bits to represent each dimension. They have shown that binarization can be comparable to or even better than the original word embeddings when used as features for named entity recognition tasks. Moreover, Faruqui et al. (Faruqui et al., 2015) showed that imposing sparsity constraints over the embedding vectors can further improve the representation interpretability and performance on several word similarity and text classiﬁcation bench mark datasets. These works indicate that, for some tasks, we do not need all the information encoded in “standard” word embeddings. Nonetheless, it is clear that binarization loses a lot of information, and this calls for a systematic comparison of how many bits are needed to maintain the expressivity needed from word embeddings for different tasks.

如果我们考虑传统的簇编码字表示，例如布朗簇（Brown等人，1992），它仅使用少量的位来跟踪字簇的分层树上的路径来表示每个字。事实上，词向量概况了在语言模型中连续向量表示的离散聚类表示的思想，目的是改进连续词 类比预测和泛化能力（Bengio等人，2003; Mikolov等人，2013a; Mikolov等人et al。，2013b）。然而，已经证明，作为离散特征的布朗集群甚至比连续词向量作为命名实体识别任务的特征更好（Ratinov和Roth，2009）。 Guo et al。 （Guo等人，2014）进一步尝试使用针对每个维度调整的阈值二值化嵌入，并且基本上使用小于两个位来表示每个维度。他们已经表明，当用作命名实体识别任务的特征时，二值化的效果可以与使用原始词向量相当或甚至更好。此外，Faruqui et al。 （Faruqui等人，2015）表明，对embedding vectors（内置向量）施加稀疏约束可以进一步提高在几个单词相似性和文本分类标准数据集上的表示可解释性和性能。这些工作表明，对于一些任务，我们不需要所有的信息编码在“标准”的词向量中。然而，很明显，（binarization）二值化丢失了很多信息，这就需要系统地比较对于不同的任务，需要多少比特的词向量才能保持所需的表达性。

3 Value Truncation

In this section, we introduce approaches for word embedding when the memory is limited. We truncate any value x in the word embedding into an n bit representation.

3值截断

在本节中，介绍当内存有限时词向量的方法。 我们将词向量中的任何值x截断为n位表示。

3.1 Post-processing Rounding

When the word embedding vectors are given, the most intuitive and simple way is to round the numbers to their n-bit precision. Then we can use the truncated values as features for any tasks that word embedding can be used for. For example, if we want to round x to be in the range of [−r,r], a simple function can be applied as follows.

3.1后处理舍入

对于给定的一个词向量，最直观最简单的方法是将数字四舍五入到它们的n位精度。这样就可以使用截断的近似值，作为任何使用词向量的任务的特征。例如，如果我们要将x舍入到[-r，r]的范围内，则可以应用如下的简单函数。

For example, if we want to use 8 bits to represent any value in the vectors, then we only have 256 numbers ranging from -128 to 127 for each value. In practice, we ﬁrst scale all the values and then round them to the 256 numbers.

例如，如果我们想使用8位来表示向量中的任何值，则对于每个值，我们只有256个数字，范围从-128到127。 在实践中，我们首先缩放所有的值，然后舍入到256个数字。

3.2 Training with Limited Memory

When the memory for training word embedding is also limited, we need to modify the training algorithms by introducing new data structures to reduce the bits used to encode the values. In practice, we found that in the stochastic gradient descent (SGD) iteration in word2vec algorithms (Mikolov et al., 2013a; Mikolov et al., 2013b), the updating vector’s values are often very small numbers (e.g., < 10−5). In this case, if we directly apply the rounding method to certain precisions (e.g., 8 bits), the update of word vectors will always be zero. For example, the 8-bit precision is 2−7 = 0.0078, so 10−5 is not signiﬁcant enough to update the vector with 8-bit values. Therefore, we consider the following two ways to improve this.

Stochastic Rounding. We ﬁrst consider using stochastic rounding (Gupta et al., 2015) to train word embedding. Stochastic rounding introduces some randomness into the rounding mechanism, which has been proven to be helpful when there are many parameters in the learning system, such as deep learning systems(Guptaetal.,2015). Here we also introduce this approach to update word embedding vectors in SGD. The probability of rounding x to [x] is proportional to the proximity of x to [x]:

当训练词向量的存储器有限时，我们需要引入新的数据结构，来减少用于对值进行编码的比特位数，来修改训练算法。实际上，我们发现在随机梯度下降（SGD）迭代中使用word2vec算法（Mikolov等人，2013a; Mikolov等人，2013b），更新向量的值通常是非常小的数字（例如，<10-5 ）。在这种情况下，如果我们直接将舍入方法应用于某些精度（例如，8比特），则词向量的更新将总是为零。例如，8位精度为2-7 = 0.0078，因此10-5不足以用8位值更新向量。因此，我们考虑以下两种方法来改善这一点。

随机舍入。我们首先考虑使用随机舍入（Gupta等人，2015）来训练词向量。随机舍入将一些随机性引入舍入机制，已经证明，当学习系统中有许多参数，如深度学习系统（Guptaetal。，2015）时，这是有帮助的。在这里我们还介绍这种方法来更新SGD中的字嵌入向量。将x取整为[x]的概率与x与[x]的接近度成比例：

In this case, even though the update values are not signiﬁcant enough to update the word embedding vectors, we randomly choose some of the values being updated proportional to the value of how close the update value is to the rounding precision. Auxiliary Update Vectors. In addition to the method of directly applying rounding to the values, we also provide a method using auxiliary update vectors to trade precision for more space. Suppose we know the range of update value in SGD as [−r0,r0], and we use additional m bits to store all the values less than the limited numerical precision . Here r0 can be easily estimated by running SGD for several examples. Then the real precision is.

For example, if r0 = 10−4 and m = 8, then the numerical precisionis7.8·10−7 which can capture much higher precision than the SGD update values have. When the cumulated values in the auxiliary update vectors are greater than the original numerical precision , e.g., = 2−7 for 8 bits, we update the original vector and clear the value in the auxiliary vector. In this case, we can have ﬁnal n-bit values in word embedding vectors as good as the method presented in Section 3.1.

在这种情况下，即使更新值不足以更新字嵌入向量，我们随机地选择一些被更新的值与更新值与舍入精度接近的值成比例。

辅助更新向量。除了直接对值应用舍入的方法之外，我们还提供使用辅助更新向量来交易精度以获得更多空间的方法。假设我们知道SGD中的更新值的范围为[-r0，r0]，并且我们使用附加的m位来存储小于有限数值精度的所有值。这里r0可以很容易地通过运行SGD几个例子估计。然后真实精度为。

例如，如果r0 = 10-4和m = 8，则数值精度为7.8·10-7，其可以捕获比SGD更新值高得多的精度。当辅助更新向量中的累加值大于原始数值精度（例如对于8位为= 2-7）时，我们更新原始向量并清除辅助向量中的值。在这种情况下，我们可以在字嵌入向量中有最后的n位值与第3.1节中介绍的方法一样好。

4 Experiments on Word/Phrase Similarity

In this section, we describe a comprehensive study on tasks that have been used for evaluating word embeddings. We train the word embedding algorithms, word2vec(Mikolov et al., 2013a; Mikolov et al., 2013b), based on the Oct. 2013 Wikipedia dump. We ﬁrst compare levels of truncation of word2vec embeddings, and then evaluate the stochastic rounding and the auxiliary vectors based methods for training word2vec vectors.

在本节中，描述了一个用于评估词向量的任务的综合研究。 我们训练词向量算法word2vec（Mikolov et al。，2013a; Mikolov等人，2013b），基于2013年10月维基百科dump.首先比较word2vec嵌入的截断水平，然后基于训练word2vec向量的方法来估随机舍入和 基于辅助向量。

4.1 Datasets

We use multiple test datasets as follows.

Word Similarity.

Word similarity datasets have been widely used to evaluate word embedding results. We use the datasets summarized by Faruqui and Dyer (Faruqui and Dyer, 2014): wordsim-353, ordsim-sim,wordsim-rel,MC-30, RG-65, MTurk-287, MTurk-771, MEN3000, YP130,Rare-Word,Verb-143,andSimLex-999.We compute the similarities between pairs of words and check the Spearman’s rank correlation coefﬁcient (Myers and Well., 1995) between the computer and the human labeled ranks.

Paraphrases(bigrams).

We use the paraphrase (bigram) datasets used in (Wieting et al., 2015), ppdb all, bigrams vn, bigrams nn, and bigrams jnn, to test whether the truncation affects phrase level embedding. Our phrase level embedding is based on the average of the words inside each phrase. Note that it is also easy to incorporate our truncation methods into existing phrase embedding algorithms. We follow (Wieting et al., 2015) in using cosine similarity to evaluate the correlation between the computed similarity and annotated similarity between paraphrases

4.1数据集

我们使用多个测试数据集如下。

**词相似性。**

词相似性数据集已被广泛用于评价词向量结果。

我们使用由Faruqui和Dyer（Faruqui和Dyer，2014）总结的数据集：wordsim-353，wordsim-sim，wordsim-rel，MC-30，RG-65，MTurk-287，MTurk-771，MEN3000，YP130，Rare -Word，Verb-143和SimLex-999。

我们计算词对之间的相似性，并检查计算机和人类标记排名之间的斯皮尔曼等级相关系数（Myers和Well。，1995）。

**释义（双字母组或称二元语法）**。

使用释义（双字母组或称二元语法）数据集，

used in (Wieting et al.,2015), ppdb all, bigrams vn, bigrams nn, and bigrams jnn,

用于测试截断是否影响语句级嵌入（应该是语句级的向量）。我们的短语级向量基于每个短语内的单词的平均值。注意，也很容易将我们的截断方法合并到现有的短语嵌入算法中。我们遵循（Wieting等人，2015）使用余弦相似性来评估计算的相似性和注释之间的相似性

4.2 Analysis of Bits Needed

We ran both CBOW and skipgram with negative sampling (Mikolov et al., 2013a; Mikolov et al., 2013b) on the Wikipedia dump data, and set the window size of context to be ﬁve. Then we performed value truncation with 4 bits, 6 bits, and 8 bits. The results are shown in Fig.1, and the numbers of the averaged results are shown in Table 1. We also used the binarization algorithm (Guo et al., 2014) to truncate each dimension to three values; these experiments are is denoted using the sufﬁx “binary” in the ﬁgure. For both CBOW and skipgram models,we train the vectors with25and 200 dimensions respectively.

我们在维基百科转储数据上使用负样本（Mikolov等人，2013a; Mikolov等人，2013b）运行CBOW和skipgram，并将上下文的窗口大小设置为五。 然后我们用4比特，6比特和8比特执行值截断。 结果如图1所示，平均结果的数字如表1所示。我们还使用二值化算法（Guo et al。，2014）将每个维度截断为三个值; 这些实验使用图中的“二进制”表示。 对于CBOW和skipgram模型，我们分别训练25和200维的向量。

The representations used in our experiments were trained using the whole Wikipedia dump. A ﬁrst observation is that, in general, CBOW performs better than the skipgram model. When using the truncation method, the memory required to store the embedding is signiﬁcantly reduced, while the performance on the test datasets remains almost the same until we truncate down to 4 bits. When comparing CBOW and skipgram models, we again see that the drop in performance with 4bit values for the skipgram model is greater than the one for the CBOW model. For the CBOW model, the drop in performance with 4-bit values is greater when using 200 dimensions than it is when using 25 dimensions. However, when using skipgram, this drop is slightly greater when using 25 dimensions than 200.

我们的实验中使用的表示是使用整个维基百科转储训练。 第一个观察是，一般来说，CBOW执行比skipgram模型更好。 当使用截断方法时，存储嵌入所需的存储器显着减少，而测试数据集上的性能保持几乎相同，直到我们截断到4位。 当比较CBOW和skipgram模型时，我们再次看到，对于skipgram模型，4位值的性能下降大于CBOW模型的性能下降。 对于CBOW模型，使用200维时的4位值的性能下降比使用25维时的性能下降更大。 但是，使用跳跃图时，使用25个尺寸时，此下降稍大于200。

We also evaluated the binarization approach (Guo et al., 2014). This model uses three values, represented using two bits. We observe that, when the dimension is 25, the binarization is worse than truncation. One possible explanation has to do merely with the size of the space; while 325 is much larger than the size of the word space, it does not provide enough redundancy to exploit similarity as needed in the tasks. Consequently, the binarization approach results in worse performance. However, when the dimension is 200, this approach works much better, and outperforms the 4-bit truncation. In particular, binarization works better for skipgram than for CBOW with 200 dimensions. One possible explanation is that the binarization algorithm computes, for each dimension of the word vectors, the positive and negative means of the values and uses it to split the original values in that dimension, thus behaving like a model that clusters the values in each dimension. The success of the binarization then indicates that skipgram embeddings might be more discriminative than CBOW embeddings.

我们还评估了二值化方法（Guo et al。，2014）。此模型使用三个值，使用两个位表示。我们观察到，当维度为25时，二进制化比截断更糟。一个可能的解释仅仅是空间的大小;虽然325比字空间的大小大得多，但它不能提供足够的冗余来利用任务中所需的相似性。因此，二值化方法导致更差的性能。但是，当维度为200时，此方法工作得更好，并且优于4位截断。特别地，二进制化对于skipgram比对于具有200维的CBOW工作更好。一种可能的解释是二进制化算法为单词向量的每个维度计算值的正和负平均值，并且使用它来分割该维度中的原始值，因此表现得像将每个维度中的值聚类的模型。二进制化的成功然后表明，跳跃嵌入可能比CBOW嵌入更有区别。

4.3 Comparing Training Methods

We compare the training methods for the CBOW model in Table 2. For stochastic rounding, we scale the probability of rounding up to make sure that small gradient values will still update the values. Both stochastic rounding and truncation with auxiliary update vectors (shown in Sec. 3.2) require 16 bits for each value in the training phase. Truncation with auxiliary update vectors ﬁnally produces 8-bit-value based vectors while stochastic rounding produces 16-bit-value based vectors. Even though our auxiliary update algorithm uses smaller memory/disk to store vectors, its performance is still better than that of stochastic rounding. This is simply because the update values in SGD are too small to allow the stochastic rounding method to converge. Auxiliary update vectors achieve very similar results to the original vectors, and, in fact, result in almost the same vectors as produced by the original truncation method.

4.3比较训练方法

我们比较表2中的CBOW模型的训练方法。对于随机舍入，我们缩放向上舍入的概率，以确保小梯度值仍然更新值。 随机舍入和具有辅助更新向量的截断（在3.2节中示出）在训练阶段中对于每个值需要16个比特。 具有辅助更新向量的截断最终产生基于8位值的向量，而随机舍入产生基于16位值的向量。 即使我们的辅助更新算法使用较小的内存/磁盘来存储向量，其性能仍然优于随机舍入。 这是因为SGD中的更新值太小而不允许随机舍入方法收敛。 辅助更新向量实现与原始向量非常相似的结果，并且实际上导致与原始截断方法产生的向量几乎相同的向量。

5 Experiments on Dependency Parsing

We also incorporate word embedding results into a downstream task, dependency parsing, to evaluate whether the truncated embedding results are still good features compared to the original features. We follow the setup of (Guo et al., 2015) in a monolingual setting. We train the parser with 5,000 iterations using different truncation settings for word2vec embedding. The data used to train and evaluate the parser is the English data in the CoNLL-X shared task (Buchholz and Marsi, 2006). We follow (Guo et al., 2015) in using the labeled attachment score(LAS)to evaluate the different parsing results. Here we only show the word embedding results for 200 dimensions, since empirically we found 25-dimension results were not as stable as 200 dimensions.

5依赖性分析的实验

我们还将字嵌入结果并入下游任务，依赖解析，以评估截断嵌入结果是否仍然是与原始特征相比的良好特征。 我们遵循（Guo等人，2015）在单语环境中的设置。 我们使用不同的截断设置为word2vec嵌入训练解析器5000次迭代。 用于训练和评估解析器的数据是CoNLL-X共享任务中的英语数据（Buchholz和Marsi，2006）。 我们遵循（Guo等人，2015）在使用标记的附着分数（LAS）来评估不同的解析结果。 这里我们只显示200维的词嵌入结果，因为根据经验，我们发现25维结果不如200维稳定。

The results shown in Table 3 for dependency parsing are consistent with word similarity and paraphrasing. First, we see that binarization for CBOW and skipgram is again better than the truncation approach. Second, for truncation results, more bits leads to better results. With 8-bits, we can again obtain results similar to those obtained from the original word2vec embedding.

表3中显示的依赖性解析的结果与单词相似性和释义一致。 首先，我们看到CBOW和skipgram的二值化再次比截断方法更好。 第二，对于截断结果，更多的位导致更好的结果。 使用8位，我们可以再次获得类似于从原始word2vec嵌入获得的结果。

6 Conclusion

We systematically evaluated how small can the representation size of dense word embedding be before it starts to impact the performance of NLP tasks that use them. We considered both the ﬁnal size of the size we provide it while learning it. Our study considers both the CBOW and the skipgram models at 25 and 200 dimensions and showed that 8 bits per dimension(and sometimes even less)are sufﬁcient to represent each value and maintain performance on a range of lexical tasks. We also provided two ways to train the embeddings with reduced memory use. The natural future step is to extend these experiments and study the impact of the representation size on more advanced tasks.

6结论

我们系统地评估密集字嵌入的表示大小在开始影响使用它们的NLP任务的性能之前有多小。我们考虑了我们在学习它时提供的大小的最大尺寸。我们的研究考虑了CBOW和在25和200维度的skipgram模型，并且表明每维度8位（有时甚至更少）足以表示每个值并且在一系列词汇任务上保持性能。我们还提供了两种方法来减少内存使用的嵌入。自然的未来步骤是扩展这些实验，并研究表示大小对更高级任务的影响。

Acknowledgment

The authors thank Shyam Upadhyay for his help with the dependency parser embeddings results, and Eric Horn for his help with this write-up. This work was supported by DARPA under agreement numbers HR0011-15-2-0025 and FA8750-13-20008. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the ofﬁcial policies or endorsements, either expressed or implied, of any of the organizations that supported the work.

确认

作者感谢Shyam Upadhyay对依赖解析器嵌入结果的帮助，以及Eric Horn为这篇文章的帮助。这项工作由DARPA根据协议号HR0011-15-2-0025和FA8750-13-20008支持。美国政府有权复制和分发用于政府目的的转载，尽管其中有任何版权符号。本文包含的观点和结论是作者的观点和结论，不应被解释为必然代表支持该工作的任何组织的明示或暗示的官方政策或背书。

Table 1: The detailed average results for word similarity and paraphrases of Fig.1

表1：图1的单词相似性和释义的详细平均结果

Table 2: Comparing the training CBOW models: We set the average value of the original word2vec embeddings to be 1, and the values in the table are relative to the original embeddings baselines. “avg. (w.)” represents the average values of all word similarity datasets. “avg. (b.)” represents the average values of all bigram phrase similarity datasets. “Stoch. (16 b.)” represents the method using stochastic rounding applied to 16-bit precision. “Trunc. (8 b.)” represents the method using truncation with 8-bit auxiliary update vectors applied to 8-bit precision.

表2：比较训练CBOW模型：我们将原始word2vec嵌入的平均值设置为1，并且表中的值相对于原始嵌入基线。 “avg。 （w。）“表示所有词相似性数据集的平均值。 “avg。 （b。）“表示所有双字母相似度数据集的平均值。 “Stoch。 （16b。）“表示使用应用于16位精度的随机舍入的方法。 “Trunc。 （8b。）“表示使用应用于8位精度的8位辅助更新向量的截断的方法。

Table 3: Evaluation results for dependency parsing(inLAS).

表3：依赖性解析的评估结果（inLAS）。